Approximating Artificial Neural Network Inference With Maximum Inner Product Search

Yonatan Medan

October 10, 2021

Abstract

In this project I show that by estimating which are the top k largest activations of A Linear Neural Network layer with MIPS (Maximum Inner Product Search). We can estimate the network predictions with a small fraction of calculations, and par accuracy. Our Experiments show that by using only 5% of the most active neurons, we achive 0.9659 accuracy on MNIST dataset, compered to 0.9694 with all neurons active regaining 0.9964 of the full Network accuracy. while similar approaches are used to speed up training [1], As far as we know no study was done on approximating inference of NN by astimating the top k largest activations.

1 Introduction

Neural Networks are wide spread and are the workhorse for Modern Machine Learning Algorithms, Deep Learning Models can have up to Billions of parameters and demand large computational resources for training and inference. thus finding ways to approximate NN predictions by using less resources can help leverage large models with lower resources.
In this work I use recent methods for Approximate Maximum Inner Product search to find the k-largest Neuron Activations of a Feed Forward Layer, corresponding to the input query.
Formally, let $W \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}, \mathrm{b} \in \mathbb{R}^{\mathrm{N}}$ The weights matrix and bias vector of a Feed Forward Layer.
outputs of layer with respect to input vector $x \in \mathbb{R}^{\mathrm{M}}$ would be $y = Wx + b$. we can approximate Wx by searching for row vectors W_{Ri} in W which give the maximal inner product $W_{Ri}x$ those will be the neurons which have the largest activation. let a_k be the k largest activation and
let $TOP_K = \{i|W_{Ri}x \geq a_k\}$ then we can approximate y by:

- $y_i = W_{Ri}x + b_i$ if $i \in TOP_K$
- else $y_i = b_i$.

The same procedure can be done to find the k smallest activations, by finding the k largest activations of $-x$. let m_k be the k smallest activation and $MIN_K = \{i|W_{Ri}(-x) \geq -m_k\}$ the the final activation of the network can be updated as:

- $y_i = W_{Ri}x + b_i$ if $i \in TOP_K \cup MIN_K$
- else $y_i = b_i$

the next problem we face now is how to choose W rows that maximize the inner product $W_{Ri}x$, This can be done with methods from A-MIPS(Approximate Maximum inner product search) studies, specifically in our experiments we hash the rows of W in a way that given an input vector x, x will be mapped by a hash function to the same bucket as rows of W which yield the maximum inner product $W_{Ri}x$. these Hash fucntions Are called Asymmetric Locality Sensitive Hash functions or A-LSH.

2 Background

2.1 MIPS - Maximum Inner product search

In this problem we are given a large dataset X of n points where n is very large, and a query vector q, Our goal is to find a vector $p \in X$ which maximizes the inner product with q. formally, given a set $X = \{x_1, ...x_n\}$ of points in euqlidian space, and a query vector q. find

$$x^\star = \underset{i \in X}{\operatorname{argmax}}\ q^T x_i$$

Due to the difficulty of finding the exact maximizing vector in high dimensional space, an approximated verion of the problem can be formed.
given approximization ratio c ($0 < c < 1$), and a query q. find $x_i \in X$ s.t $q^T x_i \geq c q^T x^\star$ this problem is calls c-AMIPS (c- Approximate MIPS).

2.2 LSH

LSH or Locality sensative hashing [2] refers to a family of hash functions which hash points which are close to each other under some similarity function, to the same bucket with high probability. and points which are far from each other to different buckets with high probability. formally: A family H is callsed $(S_0, cS_0, p1, p2)$ sensitive if, for any two points $x, y \in \mathbb{R}^{\mathrm{D}}$ h chosen uniformaly from H statisfies the following:

- if $Sim(x, y) \geq S_0$ *then* $P(h(x) = H(y)) \geq p_1$
- if $Sim(x, y) \leq cS_0$ *then* $P(h(x) = h(y)) \leq p_2$

We can concatenate K of there hash functions to create meta hash function

$$B_l(x) = [h_1(x); ...h(x)_K]$$

where when K is large, collision probability for points which are far drops. we can enhance the collision probability of close elements by hashing each point to several buckets, by using L separate buckets where each has its own hashing function $B_l(x)$ where $l \in [L]$. By using these meta hash function, we can do the following to search for Nearest Neighbors in a dataset:

1. Hash all points in the dataset to L backets.
2. given input query q search in buckets $B_l(q)$ ($l \in [L]$)
3. return the point which is closest to q in the union of L buckets

By using a smart choice K and L we can search for Near elements in a sub linear time.
few observations could be made.

1. if all points in the dataset are normalized (on the same unit sphere) then the result of Nearest Neighbors search using LSH will have approximately the maximum inner product with the query. however if the dataset is not normalized, normalizing it would give bad results for MIPS as the norm of the vector greatly effect the inner product results.
2. using lsh directly for inner product search would give bad results, as vectors which have very different norms would result on different buckets. thus if query has small norm, it would match only small norm vectors of the dataset when searching for nearest neighbors.

2.3 ALSH for MIPS

To solve the above 2 issues [4] suggested using 2 different transformations P(x),Q(x), where P(x) is used for the dataset vectors and Q(x) is used for the query vector, using these transformation prior to hashing, reduces the MIPS problem of finding maximum inner product between dataset vector x and query q, to finding the nearest neighbor of Q(q) where dataset vectors are mapped to P(x).
resulting the following:

$$\arg\underset{x \in X}{max}\ q^T x \approx \arg\underset{x \in X}{max} ||Q(q) - P(x)||$$

2.4 H2-ALSH

A very interesting and efficient variation of ALSH is the one introduced in [3],
where all points in dataset X are first divided to groups based on their norms. then given a query q. groups with greater norms are searched first by ALSH, (as those have larger probability for MIP with q).
to reduce the search in each group to Nearest Neighbor problem, each group with max norm of M is mapped by $P_M(x)$ where

$$P_M(x) = [x_1, x_2, ...x_d; \sqrt{M^2 - ||x||^2}]$$

notice $||P_M(x)||_2^2 = M$. thus all vectors are mapped to the same sphere surface. to search for group with max norm M we map query q to Q(q), where

$$Q_M(q) = [\lambda q_1, ...\lambda q_2; 0]$$

where $\lambda = \frac{M}{||q||}$. according to the above we get:

$$||Q(q) - P(x)||^2 = M^2 + \lambda^2||q||^2 - 2\lambda q^T x$$

where M is constant and Lambda is constant under query. thus we get

$$\arg\underset{x \in X}{max}\ q^T x \approx \arg\underset{x \in X}{max}||Q(q) - P(x)||$$

i.e the problem is reduced for each group to Nearest Neighbors search.

3 Contribution

Using the H2_ALSH we can construct an algorithm that approximates a neural network layer output. we first hash the layer Weight matrix W rows with H2_ALSH. then given input vector x, bias vector b and weight Matrix W.

Algorithm 1: MIPS Layer Approximation

initialize y = b;
TOP_ROW_INDEXES = H2_ALSH.findTopK (K,x);
MIN_ROW_INDEXES = H2_ALSH.findTopK (K,-x);
foreach $i \in TOP\_ROW\_INDEXES \cup MIN\_ROW\_INDEXES$ **do**
  $y_i \leftarrow W_{Ri}x + b_i$
end

Our Hypothesis is that Only a fraction of neuron activation is affecting the end result, thus approximating the K most largest and smallest activations would give near par results. in the next section we test our hypothesis empirically

4 Experiments

4.1 Approximating Neural Network predictions on MNIST

In this section we present our finding after implementing Algorithm 1 using a 2 layer neural network trained to predict Mnist digits. the first layer has weights with size 784×784, and the second 10×784, h2_alsh was applied to hash the weight matrix of the first layer rows, inference of the first layer is done according to Algorithm 1,followed by Relu activation function, 2nd layer (having only 10 rows) is computed as a regular Feed Forward layer, and final class probabilities are computed using softmax of the 2nd layer.
the results of the experiment are shown in Table 1.

Number Of Active Neurons	Active Neuron Ratio	Accuracy	Accuracy Compered to Full Activation
10	1.28%	0.8464	0.8731173922
20	2.55%	0.9411	0.9708066845
40	5.10%	0.9659	0.9963895193
80	10.20%	0.9691	0.9996905302
784	100.00%	0.9694	1

Table 1: Comparison of accuracy for different Number of active neurons.

As seen above by using 5% of the most active neurons we get results that are as 99.6% accurate as the results using all neuron activation. which confirms our hypothesis.

5 Limitations

While showing a case which approximation is very successful, a few takeaways must accounted,

- while lsh has sub-linear search speed on a large datasets we suspect it might require some constant time and space resources which are negligible on large detests but might not be as such on small dataset, as the case for the row vectors of a feed-forward NN weight matrix. thus it yet needs proving that time can be spared when using the proposed method as compered to modern optimized inference methods.
- while we showed a specific case where approximization succeds, it might not be the general case for example Mnist is somewhat sparse (most of the inputs are 0 which represents black.) so approximation for more dense datasets should be tested to confirm our hypotesis.
- In our experiments we only approximated a single layer, of a shallow neural network, more experiments can be made for much larger and sophisticated networks.

6 Conclusion and Future Work

In this project we showed a case for approximating Feed Forward layer by using a fraction of its neurons, we also proposed an algorithm to do so in an efficient way. In future work we can analyze the time and space efficiency of the proposed method, compering it with cpu and gpu inference of full network activation. further more we can examine more datasets, and more sophisticated Neural Network architectures.

References

[1] B. Chen, T. Medini, J. Farwell, S. Gobriel, C. Tai, and A. Shrivastava. Slide : In defense of smart algorithms over hardware acceleration for large-scale deep learning systems, 2020.

[2] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, page 253–262, New York, NY, USA, 2004. Association for Computing Machinery.

[3] Q. Huang, G. Ma, J. Feng, Q. Fang, and A. K. H. Tung. Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '18, page 1561–1570, New York, NY, USA, 2018. Association for Computing Machinery.

[4] A. Shrivastava and P. Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips), 2014.